IN THE SPECIFICATION

Please delete paragraphs [0023] – [0025].

Please replace paragraph [0030] with the following amended paragraph:

[0030] For illustrative purposes, several embodiments of the invention are disclosed below in the context of a blade server environment. As an overview, typical blade server components and systems for which resource sharing schemes in accordance with embodiments of the invention may be generally implemented are shown in Figures 1a-c and 2. Under a typical configuration, a rack-mounted chassis 100 is employed to provide power and communication functions for a plurality of blades 102, each of which occupies a corresponding slot. (It is noted that all slots in a chassis do not need to be occupied.) In turn, one [[of]] <u>or</u> more chassis 100 may be installed in a blade server rack 103 shown in Figure 1c. Each blade is coupled to an interface plane 104 (*i.e.*, a backplane or mid-plane) upon installation via one or more mating connectors. Typically, the interface plane will include a plurality of respective mating connectors that provide power and communication signals to the blades. Under current practices, many interface planes provide "hot-swapping" functionality – that is, blades can be added or removed ("hot-swapped") on the fly, without taking the entire chassis down through appropriate power and data signal buffering.

Please replace paragraph [0032] with the following amended paragraph:

[0032] An important feature required of all blade servers is the ability to ~~communication~~ communicate externally with other IT infrastructure. This is typically facilitated via one or more network connect cards 110, each of which is coupled to interface plane 104. Generally, a network connect card may include a physical interface comprising a plurality of network port connections (*e.g.*, RJ-45 ports), or may comprise a high-density connector designed to directly connect to a network device, such as a network switch, hub, or router.

Please replace paragraph [0055] with the following amended paragraph:

[0055] Under principles of an embodiment of the invention, an OS-transparent out-of-band communication scheme is employed to allow various types of resources to be shared across server nodes. At the same time, firmware-based components (*e.g.*, firmware drivers and API's) are employed to facilitate low-level access to the resources and rerouting of data over the OOB channel. The scheme may be effectuated across multiple computing platforms, including groups of blades, individual chassis, racks, or groups of racks. During system initialization, firmware provided on each platform is loaded and executed to set up the OOB channel and appropriate resource access and data re-routing mechanisms. Each blade then transmits information about its shared resources over the OOB to a global resource manager. The global resource manager aggregates the data and configures a "virtual" global resource. Global resource configuration data in the form of global resource descriptors then sent back to the blades to apprise the blades of the configuration and access mechanism for the global resource. Drivers are then

configured to support access to the global resource. Subsequently, the global resource descriptors are handed off to the operating system during OS load, wherein the virtual global resource appears as a local device to the operating system, and thus is employed as such during OS runtime operations without requiring any modification to the OS code. Flowchart operations and logic according to one embodiment of the process are shown in Figures 5 and 7, while corresponding operations and interactions between various components are schematically illustrated in Figures 6, 8a, and 8b.

Please replace paragraph [0073] with the following amended paragraph:

[0073] During OS runtime, global resources are ~~access~~ <u>accessed</u> via a combination of the operating system and firmware components configured to provide "low-level" access to the shared resource. Under modern OS/Firmware architectures, the device access scheme is intentionally abstracted such that the operating system vendor is not required to write a device driver that is specific to each individual device. Rather, these more explicit access details are provided by corresponding firmware device drivers. One result of this architecture is that the operating system may not directly access a hardware device. This proves advantageous in many ways. Most notably, this means the operating system does not need to know the particular low-level access configuration of the device. Thus, "virtual" resources that aggregate the resources of individual devices may be "built," and corresponding access to such devices may be abstracted through appropriately-configured firmware drivers, whereby the OS thinks the virtual resource is a real local device.

Please replace paragraph [0079] with the following amended paragraph:

[0079] Once the resource target is identified, OOB communication operations are ~~preformed~~ performed to pass the resource access request to the resource target. First, the management/access driver on the requesting platform (*e.g.*, 802-1) asserts an SMI to activate that platforms local OOB communications handler 604-1. In response, the processor on BLADE 1 switches its mode to SMM in a block 708 and dispatches its SMM handlers until OOB communication handler 604-1 is launched. In response, the OOB communication handler asserts an SMI signal on the resource target host (BLADE 2) to initiate OOB communication between the two blades. In response to the SMI, the processor mode on BLADE 2 is switched to SMM in a block 710, launching its OOB communication handler. At this point, Blades 1 and 2 are enabled to communicate via OOB channel 610, and the access request is received by OOB communications handler 604-2. After the resource access request has been sent, in one embodiment an "RSM" instruction is issued to the processor on BLADE 1 to switch the processor's operating mode back to what it was before being switched to SMM.

Please replace paragraph [0080] with the following amended paragraph:

[0080] In a block 712 the access request is then passed to management/access driver 802-2 via its API. In an optional embodiment, a query is then performed in a block 714 to ~~verity~~ verify that the platform receiving the access request is the actual host of the target resource. If it isn't the correct host, in one embodiment a message

is passed back to the requester indicating so (not shown). In another embodiment, [[am]] an appropriate global resource manager is apprised of the situation. In essence, this situation would occur if the local global resource maps contained different information (*i.e.*, are no longer synchronized). In response, the global resource manager would issue a command to resynchronize the local global resource maps (all not shown).

Please replace paragraph [0083] with the following amended paragraph:

[0083] A similar resource access process is performed using a single global resource map in place of the local copies of the global resource map in the embodiment of Figure 8b. In short, many of the operations are the same as those ~~discusses~~ discussed above with reference to Figure 8a, except that global resource manager 608 is employed as a proxy for accessing the resource, rather than using local global resource maps. Thus, the resource access request is sent to global resource manager 608 via OOB channel 610 rather than directly to an identified resource target. Upon receipt of the request, a lookup of global resource map 812a is performed to determine the resource target. Subsequently, the data request is sent to the identified resource target, along with information identifying the requester. Upon receiving the request, the operations of blocks 712 – 728 are ~~preformed~~ performed, with the exception of optional operations 714.

Please replace paragraph [0084] with the following amended paragraph:

[0084] Each of the foregoing schemes offers their own advantages. When local global resource maps are employed, there is no need for a proxy, and thus there is [[no]] not need to change any software components operating on any of the blade server components. However, there should be a mechanism for facilitating global resource map synchronization, and the management overhead for each blade is increased. The primary advantage of employing a single global resource manager is that the synchronicity of the global resource map is ensured (since there is only one copy), and changes to the map can be made without any complicity required of the individual blades. Under most implementations, the main drawback will be providing a host for the global resource manager functions. Typically, the host may be a management component or one of the blades (*e.g.*, a nominated or default-selected blade).

Please replace paragraph [0086] with the following amended paragraph:

[0086] In general, global resource mapping data may be stored in either system memory or as firmware variable data. If stored as firmware variable data, the mapping data will be able to persist across platform shutdowns. In one embodiment, the mapping data are stored in a portion of system memory that is hidden from the operating system. This hidden portion of system memory may include a portion of SMRAM or a portion of memory reserved by firmware during pre-boot operations. Another way to persist global resource mapping data across shutdowns is to store the data on a persistent storage device, such as a disk drive. However, when

employing a disk drive it is recommended that the mapping data are stored in a manner that is inaccessible to the platform operating system, such as in the host protected area (HPA) of the disk drive. When global resource mapping data are stored in a central repository (*i.e.*, as illustrated by the embodiment of Figure 8b), various storage options similar to those presented above may be employed. In cases in which the global resource manager is hosted by a component other than the plurality of server blades (such as hosted by management card 112 or an external management server), disk storage may be safely implemented since these hosts are not accessible by the operating systems running on the blades.

Please replace paragraph [0087] with the following amended paragraph:

[0087] A more specific implementation of resource sharing is illustrated in Figures 9a-b and 10a-b. In these cases, the resource being shared ~~comprise~~ <u>comprises</u> disk drives 218. In the embodiment 900 illustrated in Figures 9a and 10a, the storage resources provided by a plurality of disk drives 218 are aggregated to form a virtual storage volume "V:" For clarity, the storage resources for each of the disk drives is depicted as respective groups of I/O storage comprising 10 blocks. Furthermore, each of Blades 1-16 are depicted as hosting a single disk drive 218; it will be understood that an actual implementations each blade may host 0-*N* disk drives (depending on its configuration), that the number of blocks for each disk drive may vary, and that the actual number of blocks will be several orders of magnitude higher than those depicted herein.

Please delete paragraphs [0095] – [00104].